UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/786,843	02/25/2004	Jose German Rivera	82177139	2936

22879 7590 01/25/2012

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
3404 E. Harmony Road
Mail Stop 35
FORT COLLINS, CO 80528

EXAMINER	
WEI, ZHENG	
ART UNIT	**PAPER NUMBER**
2192	

NOTIFICATION DATE	DELIVERY MODE
01/25/2012	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

JERRY.SHORMA@HP.COM
ipa.mail@hp.com
laura.m.clark@hp.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte JOSE GERMAN RIVERA and LILLIAN CHOU

Appeal 2009-014652
Application 10/786,843
Technology Center 2100

Before JEAN R. HOMERE, ST. JOHN COURTENAY III, and ANDREW J. DILLION, *Administrative Patent Judges*.

COURTENAY, *Administrative Patent Judge*.

DECISION ON APPEAL

STATEMENT OF THE CASE

This is an appeal under 35 U.S.C. § 134(a) from the Examiner's final rejection of claims 1, 4-11, 14-21, 24-31, 34-41, and 44-49, which are the remaining claims in the application. Claims 2, 3, 12, 13, 22, 23, 32, 33, 42 and 43 were cancelled during prosecution. We have jurisdiction under 35 U.S.C. § 6(b).

We Affirm.

Invention

Appellants' invention relates generally to monitoring computer software. More particularly, the invention on appeal is directed to receiving an assertion from an executing process, recording the assertion when it is violated, and allowing the executing process to continue execution. (Para. [0010]).

Illustrative Claim

1. A method for monitoring computer software comprising:

receiving an assertion from an executing process, wherein the executing process is integral to an operating system and wherein receiving an assertion comprises:

receiving an assertion request;

performing at least one of:

recognizing an assertion request type corresponding to the assertion request; or

determining a component that sourced the assertion request; and

accepting the assertion request for at least one of:

an assertion request of an enabled recognized assertion request type; or

an assertion request of a determined component which has assertion requests enabled;

recording the assertion when the assertion is violated; and

allowing the executing process to continue execution.

(emphasis added regarding disputed limitation).

Rejections

1. Claims 1, 4, 5, 7-11, 14, 15, 17-21, 24, 25, 27-31, 34, 35, 37-41, and 44-49 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over the combination of Williams ("Microsoft Visual C# .NET," May 15, 2002), GNU ("The GNU Library" Aug. 12, 1999), and PHP ("PHP Manual" May, 30, 2003).
2. Claims 6, 16, 26, and 36 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over the combination of Williams, GNU, PHP, and Cantrill (US 7,146,473, Dec. 5, 2006).

Claim Grouping

Appellants argue the obviousness rejection of claims 1, 4, 5, 7-11, 14, 15, 17-21, 24, 25, 27-31, 34, 35, 37-41, and 44-49 as a single group. (App. Br. 3-8). We select independent claim 1 as the representative claim. *See* 37 C.F.R. § 41.37(c)(1)(vii).

ISSUE (1)

> Under §103, did the Examiner err in finding that the cited combination of references would have taught or suggested "receiving an assertion from an executing process, wherein the executing process is integral to an operating system, within the meaning of claim 1?

ANALYSIS

Appellants contend that neither Williams nor the GNU reference disclose or suggest "receiving an assertion from an executing process, wherein the executing process is integral to an operating system" in the entirety. (App. Br., sec. VII A. 1). To the extent that Appellants are contending that neither reference teaches the *entire* claim limitation, we observe that the Examiner relied on Williams to teach or suggest "receiving

an assertion from an executing process" (Ans. 4), and GNU to teach or suggest that the "executing process is *integral to* an operating system." (Ans. 4). Therefore, we do not find persuasive Appellants' singular attacks on Williams and GNU.[1]

Appellants also contend that GNU fails to disclose receiving an assertion from an executing process *integral to an operating system*. (App. Br. sec. VII A. 2).[2] In particular, Appellants contend:

> The PTO asserts that the *"GNU* reference is used to teach the limitation of 'the executing process is integral to an operating system.'" The PTO is in error and has not responded to Appellants arguments showing that *GNU* fails to disclose receiving an assertion from an executing process *integral to* an operating system. *GNU* appears to describe using an assert macro to check for "an error return from an operating system function" and not receipt of an assertion *from an operating system*. The error return, as described, causes the program assertion to fail and abort, but there is no disclosure of an assertion *from the* operating system.
>
> Further, *GNU* states that on an assertion being given invalid input, a program will abort. See page 2, penultimate paragraph ("your program should not abort when given invalid input, as assert would do").
>
> (*Id.*). (emphasis added).

---

[1] *See In re Merck & Co., Inc.*, 800 F.2d 1091, 1097 (Fed. Cir. 1986) (one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references.).

[2] Appellants' principal Brief contains no page numbers. Appellants are advised to heed such formalities in the future.

The Examiner concludes as a matter of claim construction that the plain language of the claim does not require that the assertion is *from* the operating system as argued by Appellants (*Id.*), only that the executing process is *integral to* an operating system. (Ans. 14, *see also* claim 1). We agree.

Regarding the disputed claimed phrase "integral to an operating system," we begin our analysis by looking to Appellants' Specification for *context.* Appellants' Specification discloses that "where a process is, for example, integral to an operating system, the operation system will not crash the system." (Spec. para. [0012])(underline added).

Since the exemplary language given in Appellants' Specification does not give notice to an artisan of a specific definition, and cannot be read into the claims, we accord the claim term "integral to" its broadest reasonable interpretation consistent with the Specification. We particularly observe that Appellants' Specification at para [0012] further informs the artisan that "[t]he present method is applicable to many various types of software and the *scope of the appended claims is not intended to be limited to examples herein cited; e.g. such as an operating system or a user application*." (emphasis added).

The Examiner construed the aforementioned phrase as "an assertion from a process of a program running or executing under (integral to) the operating system[,] which includes any assertions from any executing processes of an application program." (Ans. 14). Given the exemplary non-limiting language found in Appellants' Specification at paragraph [0012], on this record we are not persuaded that the Examiner's claim construction is overly broad, unreasonable, or inconsistent with the Specification.

Given the aforementioned claim construction, we agree with and adopt the Examiner's findings regarding the disputed limitation: "receiving an assertion from an executing process *integral to an operating system*." (Ans. 5; *see also* claim 1).

As pointed out by the Examiner, GNU discloses:

> When you're writing a program, it's often a good idea to put in checks at strategic places for "impossible" errors or violations of basic assumptions. These kinds of checks are helpful in debugging problems with the interfaces between different parts of the program, for example.

(GNU pg. 1).

In addition, GNU discloses:

> Sometimes the "impossible" condition you want to check for is an error return from an operating system function. Then it is useful to display not only where the program crashes, but also what error was returned. The assert_perror macro makes this easy.

(GNU pg. 2).

Thus, the first portion of GNU discusses putting in checks for "impossible errors." (GNU 1). The second portion discusses a particular type of "impossible" condition, in the program being written, which is an error return from an *operating system function* that was called by the program executing within the context of a process. (GNU 2).

Based on the two aforementioned findings, we agree with the Examiner that GNU at least strongly suggests an executing process that is "integral to" (i.e., running under the control of) an operating system, within the broad scope of claim 1.

In the Reply Brief, Appellants restate that GNU teaches checking for an error from an operating system and not integral to an operating system. (Reply Br. 4). Appellants next attempt to define "integral to" according to an unidentified dictionary definition (as meaning "part of"), and by the aforementioned example found in paragraph [0012] of Appellants' Specification. (Reply Br. 4). Given that Appellants have not provided a specific named dictionary source (XI. Evidence Appendix – "None."), and in light of the express non-limiting language found in Appellants' Specification at paragraph [0012], we find Appellants' arguments unpersuasive of Examiner error.

Moreover, Appellants could have amended their claims during prosecution to recite "*part of*" instead of "*integral to*" to clarify the meaning and intended scope of the disputed claim limitation "an executing process *integral to* an operating system." (Claim 1). However, Appellants chose not to do so.[3] Because "applicants may amend claims to narrow their scope, a broad construction during prosecution creates no unfairness to the applicant or patentee." *In re ICON Health and Fitness, Inc.*, 496 F.3d 1374, 1379 (Fed. Cir. 2007) (citing *In re Am. Acad. of Sci. Tech Ctr.*, 367 F.3d 1359, 1364 (Fed. Cir. 2004)).

In particular, Appellants have not persuaded us that an error code returned by an operating system call that is trapped by a calling program that

[3] The claim as a whole must be considered to determine whether it apprises one of ordinary skill in the art of its scope, and therefore serves the notice function required by 35 U.S.C. § 112, second paragraph by providing clear warning to others as to what constitutes the infringement of the patent. *Solomon v. Kimberly-Clark Corp.*, 216 F.3d 1372, 1379 (Fed. Cir. 2000).

is being written and debugged, as discussed in GNU (pages 1-2), is not an executing process that is "integral to" an operating system, or at least suggestive of Appellants' claimed subject matter.

For these reasons, we are not persuaded that the Examiner erred in finding that the cited combination of references would have taught or suggested "receiving an assertion from an executing process, *wherein the executing process is integral to an operating system,*" within the meaning of claim 1. (emphasis added).

ISSUE (2)

Under § 103, did the Examiner err in finding that Williams, GNU, and PHP would have taught or suggested *allowing the executing process to continue execution*, within the meaning of claim 1?

ANALYSIS

Appellants contend that the Examiner relied on GNU to teach or suggest *allowing the executing process to continue execution*, (App. Br. sec. VII A. 3). We disagree for the reasons discussed below.

The Examiner relied on the teachings of Williams to teach or suggest the *allowing the executing process to continue execution.* (Final Office Action, 4; *see also* Ans. 5, last paragraph referring to the Dialog Box shown in Williams' Fig. 9-3). According to Williams, a Dialog box is generated by trace and debug output. (Williams pgs. 10-11, Fig. 9-3). The dialog box includes an "Ignore" button. (Fig. 9-3). The Examiner contends that the "Ignore" button teaches or suggests continuing the execution of the process. (Ans. 5).

Appellants disagree. Appellants contend that Williams fails to definitively disclose the contents of the displayed dialog box and what happens as a result of activating one of the displayed buttons. (*See* sixth enumerated point under the "Stated Rejections from the Final Office Action" heading on an unnumbered page of the principal Brief.). *See* n.2 *supra*.

Although we agree with Appellants that Williams does not definitively disclose what happens after the "Ignore" button is activated, it is nevertheless our view that when presented with "Abort" and "Retry" buttons, Williams at least strongly suggests that activation of the "Ignore" button would result in the process continuing, as opposed to stopping (Abort) or restarting (Retry).

On this record, we are not persuaded that the Examiner erred in determining that the cited combination of references would have taught or suggested *allowing the executing process to continue execution*, within the meaning of claim 1.

ISSUE (3)

Under §103, did the Examiner err in combining GNU and Williams?

ANALYSIS

Appellants contend that the PTO failed to provide a proper motivation to combine Williams and GNU. (App. Br. sec. VII, A. 4).

The Examiner's stated rationale is reproduced below:

> Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to put the assertion in the operation system code to monitor the error from the system call. One would have been motivated to do so to display all the error information and further help to debug

> problem as suggested by GNU (see for example, p.1, section "Explicitly Checking Internal Consistency", first paragraph, "These kinds of checks are helpful in debugging problem...").
>
> (Ans. 4)

Based upon our review of the record, it is our view that the Examiner's proffered motivation consists of an articulated reasoning with sufficient rational underpinning to support the ultimate legal conclusion of obviousness.

The Supreme Court guides that the conclusion of obviousness *can* be based on the interrelated teachings of multiple patents, the effects of demands known to the design community or present in the marketplace, and the background knowledge possessed by a person having ordinary skill in the art, and an obviousness "analysis need not seek out precise teachings directed to the specific subject matter of the challenged claim, for a court can take account of the inferences and creative steps that a person of ordinary skill in the art would employ." *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 418 (2007). *See also Dystar Textilfarben GmbH & Co. Deutschland KG v. C.H. Patrick Co.*, 464 F.3d 1356, 1368 (Fed. Cir. 2006).

Here, it is our view that an artisan possessing common sense and creativity at the time of the invention would have been familiar with various methods of debugging computer programs. While we are fully aware that hindsight bias often plagues determinations of obviousness, *Graham v. John Deere Co.*, 383 U.S. 1, 36 (1966), we are also mindful that the Supreme Court has clearly stated that the "combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results," *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 401 (2007)..

We note that Appellants have not rebutted the Examiner's legal conclusion of obviousness by showing that the claimed combination of familiar elements produces any new function. Therefore, we find Appellants' arguments unavailing regarding the combinability of the cited Williams and "GNU" references for essentially the same reasons articulated by the Examiner in the Answer. (Ans. 4, 17), and as further discussed above. For these reasons, we find Appellants' arguments unpersuasive that the Examiner erred by improperly combining the cited references under § 103.

ISSUE (4)

Under §103, did the Examiner err in finding that the combination of Williams, GNU and PHP, would have taught or suggested *recognizing an assertion request type*, within the meaning of independent claim 1?

ANALYSIS

Appellants contend that the "PHP" document does not disclose *recognizing an assertion request type*. (App. Br. sec. 6). Appellants further contend that the "PTO has failed to articulate how setting/getting various assert flags corresponds to recognition of an assertion request type as claimed in claim 1." "The PTO has failed to identify any such disclosure or teaching of an 'assert request type' in PHP." (App. Br. sec. VII A. 5).

We disagree for the reasons discussed *infra*.

The Examiner cited PHP to teach an assertion request type as "Assert Options." (Ans. 17). The Examiner articulated how the cited portion of PHP corresponds to the recognition of an assertion type. (Ans. 17). We note that

Appellants have not addressed the Examiner's specific findings articulated in the Answer.

Appellants' conclusory arguments do not persuade us of Examiner error. (See App. Br. sec. VII A. 5). This form of argument is ineffective in demonstrating error in the Examiner's underlying factual findings or ultimate legal conclusion of obviousness. See *Ex parte Belinne*, No. 2009-004693, slip op. at 7-8 (BPAI Aug. 10, 2009) (informative), *available at* http://www.uspto.gov/web/offices/dcom/bpai/its/fd09004693.pdf

Based on this record, we are not persuaded that the Examiner erred in finding that the combination of Williams, GNU and PHP would have taught or suggested *recognizing an assertion request type*, within the meaning of independent claim 1.

Appellants also contend that the cited references do not teach or suggest *determining a component that sourced the assertion request*, as recited in claim 1. We note that this limitation is complementary to the "recognizing" limitation discussed *supra*. Regarding this limitation, we agree with the Examiner's responsive argument as articulated on page 18 of the Answer.

Based on this record regarding *Issues 1-4*, we sustain the Examiner's rejection of representative claim 1, and claims 4, 5, 7-11, 14, 15, 17-21, 24, 25, 27-31, 34, 35, 37-41, and 44-49, which fall therewith. *See* 37 C.F.R. § 41.37(c)(1)(vii).

Claims 6, 16, 26, and 36 – second-stated rejection

As noted above, claims 6, 16, 26, and 36 stand rejected over the combination of Willams, GNU, PHP and Cantrill. Appellants do not present

separate arguments for the patentability of these claims and instead rely on the arguments previously presented for claim 1, which we have fully addressed *supra*, and found unpersuasive. Therefore, we sustain the Examiner's obviousness rejection of claims 6, 16, 26, and 36 for the same reasons discussed above regarding independent claim 1.

DECISION

We affirm the § 103 rejections of claims 1, 4-11, 14-21, 24-31, 34-41, and 44-49.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a)(1)(iv) (2011).

ORDER

AFFIRMED

tkl